

SN3262 – Network Administration, Management & Security

Exercise 5: Use of *ssh* in scripts

ssh (Secure Shell) can be used to run commands remotely: e.g.

```
ssh XXXX "uname -a"
```

NOTE the quotation marks.

Try the above with XXXX being the name of a remote machine that is running Linux.

```
johnh@hardy:~$ ssh topeka "uname -a"
The authenticity of host 'topeka (149.170.13.45)' can't be established.
RSA key fingerprint is a8:de:e2:5c:02:08:4e:dc:37:de:27:68:d7:ca:05:59.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'topeka' (RSA) to the list of known hosts.
Linux topeka 2.6.18-4-686 #1 SMP Wed May 9 23:03:12 UTC 2007 i686 GNU/Linux
```

Note *yes* entered on line 3 of the response to the command.

```
ssh topeka "uname -a"
```

The only problem is you have to supply a password, which while secure is not ideal for use within a script where a large number of machines are to be contacted. However there is a way in *ssh* of “securely” running remote tasks without providing a password. It uses public-key or asymmetric cryptography. You need to generate two keys, a private one and a public one, using *ssh-keygen*.

```
ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/users/johnh/.ssh/id_rsa):
Created directory '/home/users/johnh/.ssh'.
Enter passphrase (empty for no passphrase): hit return at this point
Enter same passphrase again: hit return again
Your identification has been saved in /home/users/johnh/.ssh/id_rsa.
Your public key has been saved in /home/users/johnh/.ssh/id_rsa.pub.
The key fingerprint is:
d9:94:30:20:6e:18:99:0e:41:93:b1:90:67:b1:4d:6b johnh@detroit
```

The “-t rsa” option tells *ssh-keygen* to generate an “RSA” key pair. If one does not exist, *ssh-keygen* creates a “.ssh” directory. Check if it exists:

```
ls -ld .ssh
```

The “.ssh” directory created is only accessible by the owner. You need to change the permissions so that it is readable and executable by “others”. You can do this with the following command:

```
chmod 755 .ssh
```

Confirm that the permissions have changed:

```
ls -ld .ssh
```

Change into the the “.ssh” directory and have a look at the files *ssh-keygen* has created:

```
cd .ssh
```

```
ls
```

You should see two files, one will contain your private key and the other will contain your public key. Check out the permissions for the two files, do they make sense to you? Now put your public key into a file called “authorized_keys”. You can do this by copying the “id_rsa.pub” file:

```
cp id_rsa.pub authorized_keys
```

Now login to a “remote” machine in the room:

```
ssh topeka
```

```
The authenticity of host 'topeka (149.170.13.45)' can't be established.
```

```
RSA key fingerprint is 8b:18:b9:d9:45:4a:f4:04:96:e0:c4:b1:a0:aa:e3:5d.
```

```
Are you sure you want to continue connecting (yes/no)? yes
```

```
Warning: Permanently added 'topeka,149.170.13.45' (RSA) to the list of known hosts.
```

```
Last login: Tue Oct 21 16:21:55 2003 from detroit.doc.stu.mmu.ac.uk
```

When you login for the first time you will get the statement about “authenticity” and a prompt will ask you if you want to continue. Type “yes” at this point. If it has worked then you should get the shell prompt. If you are prompted for your password then something has gone wrong. I suggest you check file permissions and see if the “authorized_keys” file is spelled correctly. Check you are actually logged into the remote machine

```
uname -a
```

```
Linux topeka 2.4.20-20.7 #1 Mon Aug 18 14:56:30 EDT 2003 i686 unknown
```

Don’t worry about that “continue prompt”, *ssh* will create a “known_hosts” file in your “.ssh” directory and add the remote machine to it. Next time you login, you should drop straight to the shell without any prompting. Exit from the remote machine and attempt to login in again:

```
exit
```

```
ssh topeka
```

There should be no prompt, check that you are logged in ok:

```
uname -a
```

Log out again and then issue a “remote” command:

```
ssh topeka "uname -a"
```

Tasks

1. Use *nmap* to see which machines are up and running Linux in the lab.

```
nmap -sP 149.170.xx.yy-zz
```

where `149.170.xx.0` is the network to which the machines in the lab are attached and `yy-zz` is the range of host addresses on that network in the lab.

2. Remotely login to each machine that is up. You will get the statement about “authenticity” and a prompt will ask you if you want to continue as before except for the machine that you had previously logged into. Each machine that you remotely login to will be placed into your `known_hosts` file.
3. Now write a script using *nmap* that will:
 - (a) test to see what machines are up and running Linux in the lab and then
 - (b) remotely login to each machine that is up and print to the screen who is logged in from the X consol.

IMPORTANT

When you have finished go into the `.ssh` directory and delete every file:

```
cd .ssh  
rm *
```

The above assumes that you are starting from your home directory.